# Adaptive Noise Filtering of Image Sequences in Real Time

ALI M. REZA
U.S. Coast Guard Academy
Department of Engineering, Electrical Engineering Section
15 Mohegan Ave., New London, CT 06320
U.S.A.

*Abstract:* Filtering noise in image sequences is an important preprocessing task in many image processing applications, including but not limited to real-time x-ray image sequences obtained in angiography. The main objective in real-time noise filtering is to improve the quality of the resultant image sequences. Practically affordable approaches are generally suboptimal and deal with the spatial and temporal dimensions independently. Spatial filters can be adaptive and edge sensitive, however, they may require more hardware real estate for the real-time processing of each frame. On the other hand, temporal-only filters are one-dimensional and take advantage of temporal correlation. These 1-D temporal filters, which are applied to each individual pixel, can be designed using adaptive approaches to compensate for motions as well as noise variations. Existing adaptive 1-D filters are relatively complex and do not lend themselves to an affordable hardware implementation for real-time processing. In this article, after reviewing different filtering approaches, an adaptive temporal restoration algorithm, based on discrete Kalman filter, is developed. Adaptation in this case is with respect to the variation of the noise statistics as well as motion. In each step of the algorithm, the conventional adaptive Kalman filter proceeds if no motion is detected. However, in the case of detected motion, the adaptive Kalman filter resets itself in a way that the motion is preserved and cause no lagging in the processed image sequence. The overall procedure is suitable for hardware implementation with present FPGA/VLSI technology.

*Key–Words:* Image Sequence Filtering, Temporal and Spatial Filtering, Adaptive Filtering, Kalman Filter

## 1 Introduction

Filtering noise in image sequences has significant positive impact in real-time medical image sequence filtering. For example, this operation can improve image quality at very low dose X-ray angiography. Many different noise reduction techniques for dynamic image sequences are proposed in literature; [2]. These techniques can be divided into different categories based on their underlying assumptions and justifications. Some of these approaches are based on the assumption that the image sequences are stationary both in time and space. In these cases, the optimal filtering is done in a spatial-temporal manner and as a result, the filters are three-dimensional. These spatial-temporal filters take advantage of spatial and temporal correlations and optimally improve the image signal-to-noise ratio ($SNR$). The algorithms and hardware implementations for the 3-D filters are generally very involved and complex.

The 3-D optimal filters are generally applicable where the underlying assumptions, spatial-temporal stationary signal, is true. However, in practice, due to edges and motions, image sequences are in general non-stationary in both space and time. In addressing some of these practical problems, adaptive algorithms are proposed in which filter characteristics are adjusted based on detection of motion and/or edge in the image sequence. Adaptive 3-D spatial-temporal filters are generally very complex and prohibitive for real-time implementation.

Complexity of 3-D filters is reduced by decoupling the spatial and temporal dimensions and implementing spatial-only filters in conjunction with temporal-only filters. Extensive research has resulted in development of many different two-dimensional filters. These spatial filters are generally nonlinear and/or adaptive due to their approaches in dealing with edges in the image. On the other hand, temporal-only filters are one-dimensional and rely on temporal correlations in the image sequence. Temporal filters also need to adapt their characteristics to the motions in the image sequence and should handle different pixels accordingly. Several adaptive 1-D noise filtering algorithms exist that can be utilized for this application. However, since these techniques are generally very complex, they do not lend themselves easily to an affordable hardware implementation for real-time processing. It is important to note that these temporal filters should adapt themselves to the behavior of each individual pixel in time and hence need to perform in-

dependently from one pixel to another.

In this paper, first the problem of noise filtering in digital image sequences is discussed. Then, based on the knowledge of the problem, different practical approaches are compared and analyzed. In this consideration, both adaptive and no-adaptive techniques are considered. Our analysis include spatial-temporal, spatial-only, and temporal-only filters. Adaptation discussed here is based on motions and edges as well as noise statistics. Finally, an adaptive temporal filtering algorithm, based on Kalman formulation, is developed in a way that it adjusts its behavior with respect to the variation of the noise statistics and motion detection in the image sequence. In each step of the algorithm, the conventional adaptive Kalman filter proceeds if there is no motion. However, in the case that a motion is detected, the adaptive Kalman filter resets itself in a way that the motion is preserved and lagging is prevented.

## 2 Statement of The Problem

The problem of concern in this study is real-time restoration of image sequences in critical applications, e.g., real-time medical imaging such as X-ray angiography. The main source of degradation considered in this study is noise. This can be due to thermal electronic noise or quantum noise. Thermal noise can be modeled by a Gaussian density function with zero mean and constant variance. This noise is generally assumed to be white; almost uniformly distributed in the frequency domain. Quantum noise, however, is modeled as a signal dependent noise in which the noise variance is proportional to the signal level. Since most of the useful information in the image sequence is in the low frequency region, it seems logical that low-pass filtering of image sequences should solve this problem. This simple logical approach, however, does generate some problem due to filtering of higher frequency components which causes blurring of edges and lagging of motions.

In most cases, blurring and/or lagging problems are sever enough that conventional approaches render themselves useless. Adaptive approaches can minimize the aforementioned degradation problems. Adaptation should be based on preservation of edges and motions. In the case of spatial-only filters, we have to adapt to the transitions in the spatial domain and adjust the filter behavior accordingly. In the case of temporal-only filters, however, the adaptation should be based on the motion estimation in the image sequence. Finally, in the case of 3-D spatial-temporal filters, the adaptive algorithm should adjust the characteristics of the filter based detection of edges in the spatial domain as well as motions in the image sequence.

One important factor that should always be taken into account for adaptive filtering of image sequences is to find an adaptive algorithm with satisfactory performance (not necessarily optimal) that can be utilized for real-time implementation. This requirement is very limiting and we should make a decision as to which features to extract for adaptation and how to facilitate a proper real-time adaptation algorithm.

There is another category of filters that is referred to as robust and/or nonlinear filters. These filters are generally suboptimal but provide good solutions to the filtering problem of non-stationary signals. The class of nonlinear filters that is suitable for noise filtering of image sequences is the group of filters that are based on order statistics; [1]. The simplest filter in this group is the median filter which is robust with respect to impulsive noise and preserves sudden changes in the signal. Another class of nonlinear filters is based on linearization of the nonlinear problem and application of a linear filter to the resultant linearized problem. The most familier exmple in this case is what is referred to as extended Kalman filte; [8]. The more advanced and computationally involved class of nonlinear filters depends on fuzzy systems [13]. One such system is applied to images in [12]. A fuzzy system that behaves as a nonlinear filter and is referred to as cluster filtering, is first proposed in [5]. This filter is based on fuzzy clustering of the signal in a locality, temporal and/or spatial, and decision making process. Decisions can generally be based on crisp or fuzzy logic. Cramer-Rao lower bounds for performance of such filters is presented in [11]. The main problem with these nonlinear filters, in general, is the complexity of their algorithms and the issues related to their real-time implementations.

In the following, the problems raised in this section are specifically addressed and, when appropriate, theoretical formulation is presented. Final recommendations, based on different scenarios are made at the end, by proposing an adaptive temporal Kalman filter which resets itself whenever a motion is detected. This method works only in temporal direction and, in general, is independent from one pixel to another.

## 3 Analysis and Comparison

In this section first different conventional approaches to image sequence filtering are discussed. Then the issue of adaptation, under different circumstances, is addressed. Finally, the idea of nonlinear filtering, in general and its application to image sequences in particular, is considered. In each case, the intention is

to discuss the relevance of the proposed technique to digital processing of image sequences.

### 3.1 Spatial-Temporal Filters

Our analysis is based on the assumption that the image sequence is presented to the filtering algorithm or its hardware implementation as a three-dimensional (3D) digital signal. Each frame is represented by a two dimensional array and sequence of frames are assumed to be temporal samples in time. Generally, temporal sampling rate is fixed and when needed we consider it to be 30 frames per second or temporal sampling rate of 30 Hz. In this case, the problem can be stated mathematically as follows. The image sequence observed by the imaging system is represented by a 3D array $x(n, m, k)$, given in (1) in which $f(n, m, k)$ is the ideal noise free image sequence and $\eta(n, m, k)$ is the additive noise sequence.

$$x(n, m, k) = f(n, m, k) + \eta(n, m, k) \qquad (1)$$

The objective is to process $x(n, m, k)$ in order to obtain the best possible estimate of $f(n, m, k)$. In most cases, there is a high correlation between pixels both in spatial and temporal dimensions. However, this argument is not completely true when a pixel is close to an edge or belongs to a moving object. The best estimation can be achieved when we take advantage of both spatial and temporal correlations. When a pixel is located in a locally stationary region; e.g., pixel is not close to an edge and does not belong to a moving object, a linear three dimensional filter can maximally filter white noise by proper filter design. This approach is referred to as standard linear 3D filtering. If the 3D linear filter impulse response is $h(n, m, k)$, then the filter output is given by (2), in which $S$ is the region of support for the impulse response.

$$\hat{f}(n, m, k) = \sum_{p,q,l \in S} h(p, q, l)x(n - p, m - q, k - l) \qquad (2)$$

In general, the main requirements for these filters are to have spatially linear phase response and temporally minimal lagging effect. To this end, the 3D filter is spatially FIR and temporally IIR. Conventional approach in this case is based on minimum mean-square-error (MMSE) or Wiener filter [3]. The knowledge of the signal autocorrelation function is not generally available and, in practice, based on the assumption that the signal is stationary and noise is white, the correlation function is estimated from available data. The recursive part of the 3D filter; i.e., the temporal component, can be implemented based on Kalman filtering. This implementation is more suitable for adaptation. This separation of the spatial and temporal domains is based on the assumption that the temporal and spatial components of the 3D filter are separable.

In its general form, the 3D filter, in a non-adaptive implementation, cannot remain optimum for all pixels and at all times. The optimality can only be achieved when the 3D signal is stationary. Adaptive filters are more suitable in these circumstances. Adaptation is generally based on the idea that the 3D signal is locally stationary and the rate of filtering can be reduced properly in the vicinity of an edge and motion. The general 3D adaptive filter cannot lend itself to real-time processing and hence its complete formulation is avoided here.

When adaptation in spatial and temporal domains are separated, the real-time implementation becomes more feasible. Therefore, we first discuss spatial-only and temporal-only filters and then we address the issues of adaptation and/or robust nonlinear filtering.

### 3.2 Spatial Filters

Spatial filters for filtering noise in still images is well developed and studied; [4]. Similar approaches and algorithms can be employed for noise filtering in image sequences. Requirement of linear phase response limits our choices to the optimum FIR Wiener filter.

Let the spatial auto-correlation function of the desired noise-free image be given by $r_d(i, j)$, where $i$, and $j$, are respectively, delays in horizontal and vertical directions and $r_d(0, 0)$ is the image power. When the image is stationary, and the noise is white with variance of $\sigma_n^2$, the auto-correlation function of the noisy image becomes

$$r_n(i, j) = r_d(i, j) + \sigma_n^2 \delta(i, j) \qquad (3)$$

which is assumed to be fixed throughout the image and does not change from one pixel to another. In (3), $\delta(i, j)$ is the discrete unit impulse function. Knowledge of $r_d(i, j)$ and $\sigma_n^2$ can readily be used to design the optimum FIR Wiener filter. For simplicity and without lack of generality, we assume that the correlation function is separable, $r_d(i, j) = r_{dh}(i)r_{dv}(j)$, $r_n(i, j) = r_{nh}(i)r_{nv}(j)$ and as a result the separable optimum FIR Wiener filter can be obtained. Assuming that the filter size is $(2I + 1) \times (2J + 1)$, the horizontal and the vertical 1-D filters are given by

$$\begin{cases} \mathbf{h}_{oh} = \mathbf{R}_{nh}^{-1} \cdot \mathbf{r}_{dh} \\ \\ \mathbf{h}_{ov} = \mathbf{R}_{nv}^{-1} \cdot \mathbf{r}_{dv} \end{cases} \qquad (4)$$

where
$\mathbf{r}_{dh} = [r_{dh}(-I), r_{dh}(-I + 1), \cdots, r_{dh}(I - 1), r_{dh}(I)]^T$
$\mathbf{r}_{dv} = [r_{dv}(-I), r_{dv}(-I + 1), \cdots, r_{dv}(I - 1), r_{dv}(I)]^T$

$$\mathbf{R}_{nh} = \begin{bmatrix} r_{nh}(0) & r_{nh}(1) & \cdots & r_{nh}(2I) \\ r_{nh}(1) & r_{nh}(2) & \cdots & r_{nh}(2I-1) \\ \vdots & \vdots & \ddots & \vdots \\ r_{nh}(2I) & r_{nh}(2I-1) & \cdots & r_{nh}(0) \end{bmatrix}$$

$$\mathbf{R}_{nv} = \begin{bmatrix} r_{nv}(0) & r_{nv}(1) & \cdots & r_{nv}(2I) \\ r_{nv}(1) & r_{nv}(2) & \cdots & r_{nv}(2I-1) \\ \vdots & \vdots & \ddots & \vdots \\ r_{nv}(2I) & r_{nv}(2I-1) & \cdots & r_{nv}(0) \end{bmatrix}$$

In general, the size of the filter is chosen based on the way that the auto-correlation function dies out. For example, if we can ignore $r_d(4,4)$ in comparison with $r_d(0,0)$, then we can safely choose a filter of size $(2\times3+1)\times(2\times3+1) = 7\times7$. With this simple rule, the deviation from optimal solution is insignificant.

This 2D spatial filter is a fixed non-adaptive spatial filter that may not be useful in many practical applications. In cases where enough processing power is available, the above formulation can be adapted to the local statistics. The overall effect of this non-adaptive formulation is optimal removal of stationary noise from stationary regions of the image, with a side effect of causing some degrees of blurring on edges, where the image is not stationary.

In practice, it is desired to not only reduce the noise but also to enhance the image. This desired performance, however, cannot be achieved by application of a fixed linear spatial low-pass filter. Adaptive and robust nonlinear spatial filters; e.g [9], are heavily used in filtering of still images. Due to the inherent blurring effect of any linear fixed spatial filtering and high computational requirement for adaptive and/or nonlinear filtering, the noise filtering for image sequences is generally done in temporal domain. In the next section, this class of filters is discussed and the advantages and disadvantages of temporal filters along with some remedies for their improvements are considered.

### 3.3 Temporal Filters

In this case we discuss one-dimensional temporal filters that are more realistic for real-time implementation. This filtering approach only takes advantage of temporal correlation with no concern on existing spatial correlation in the image sequence. Due to the requirement of real-time processing and lack of linear phase requirement, it is advantages to use recursive filters. Optimality of the filter corresponds to its effectiveness in maximally filtering the noise. For simplicity, the noise characteristics, as before, can be modeled as white Gaussian random process.

For any given pixel, the variation from one frame to the next is only due to noise when there is no motion. In other words, the pixels that are located in the non-moving fields of the video image are highly correlated from one frame to another. The optimum filtering, in this case would be a very narrow-band low-pass filter which only passes the DC component. In a recursive fashion, this can simply be achieved by using

$$y(k) = \alpha y(k-1) + (1-\alpha)x(k) \qquad (5)$$

for each pixel. In this case $\alpha$ can be very close to 1 for higher averaging or filtering (note that $0 < \alpha < 1$). In (5), $k$ denotes the current frame, $y(k)$ represents the filter output (new pixel value) and $x(k)$ is the present pixel value for the current frame. Note that $y(k-1)$ denotes the filter output for the same pixel at previous frame. The corresponding frequency responses of the recursive filter given by (5), for various values of $\alpha$, are shown in Figure 1. If faster role-off is desired,

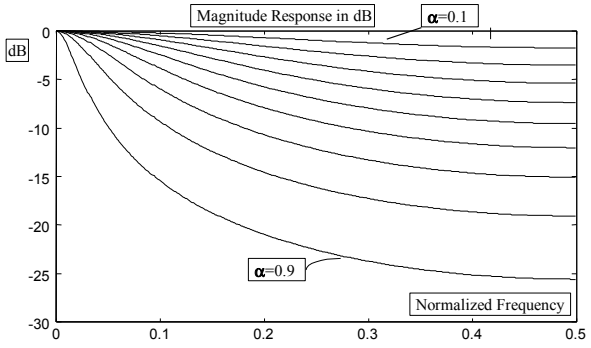


Figure 1: Frequency response of the recursive filter given in (5) for various values of $\alpha$ parameter. The best filter characteristics, with no motion, is the one associated with $\alpha = 0.9$ in these examples.

higher order recursive filters can be used. For example, by increasing the filter order by one and adding another pole at the same position; i.e., $p_1 = p_2 = \alpha$, for any given $\alpha$, the role-off is increased by a factor of two. The recursive equation, in this case, becomes

$$y(k) = 2\alpha y(k-1) - \alpha^2 y(k-2) + (1-\alpha)^2 x(k) \qquad (6)$$

which will produce better frequency response for any given $\alpha$. The corresponding frequency responses, for various values of $\alpha$, are shown in Figure 2.

The major problem with the large values of $\alpha$, in (5) and (6), is that it gives a very small weight to the pixel values on the current frame and very large weight to the previous filtered outputs. This is fine and desirable for the regions in which there is no motion. However, when there is a motion, the corresponding pixel value changes rapidly from one frame to another due to motion. Large values of $\alpha$ strongly filters rapid variations due to motion and causes a lagging effect in the image sequence. This ia an undesirable effect and

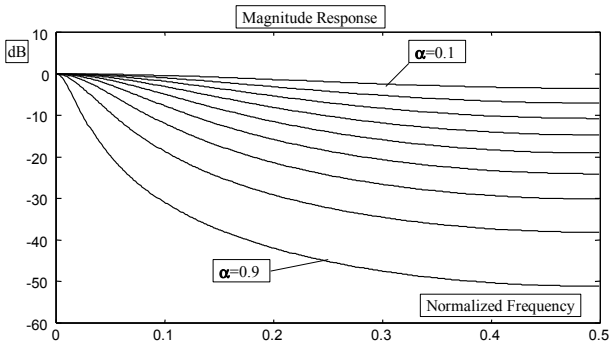


Figure 2: Frequency response of the recursive filter given in (6) for various values of $\alpha$ parameter. The best filter characteristics, with no motion, is the one associated with $\alpha = 0.9$ in these examples.

should be avoided. One way to avoid this problem, or to reduce its effect, is to decrease the value of $\alpha$. This will result in a sub-optimal performance where there is no motion. Another approach is to adjust the value of $\alpha$ in different regions of the image and not to use a fixed value at all times and for all pixels. The adjustment should be based on some form of motion estimation. This adaptive approach is addressed in the next section.

In general, the optimum choice for $\alpha$ is evaluated based on the desired noise reduction under a given condition. If there is no motion, the noise filtering can be set at its maximum acceptable level without generating any undesirable side effect. To this end, $\alpha$ should be chosen as close to 1 as possible. This choice, however, should be traded off by the maximum effective averaging that we desire to have. The relation of $\alpha$ with the effective averaging and noise filtering are derived as follows.

For the first order filter in (5), the filter's transfer function and its impulse response are respectively given by:

$$H(z) = \frac{1-\alpha}{1-\alpha z^{-1}} \text{ and } h(k) = (1-\alpha)\alpha^k u(k)$$

Comparison of the above exponential equation with the general continuous-time exponential function indicates that the system time-constant is $\tau = -T/\ln(\alpha)$ sec., where $T$ is the temporal sampling period defined as the inverse of the number of frames per second; $T = 1/f_s$. This time constant translates to $n_o = -1/\ln(\alpha)$ number of frames in the image sequence.

For calculation of the filters effective averaging length (or number of frames), it is required to find the step-response of the filter and the minimum number of frames that it takes for the output to approach its final steady-state value within a specified tolerance. If $\varepsilon_r$ is the desired tolerance, e.g., $\varepsilon_r = 0.1$ for 10% tolerance, it can be shown that the rise-time of the filter (in terms of the number of frames), is

$$n_r = \left\lceil \frac{\ln(\varepsilon_r)}{\ln(\alpha)} \right\rceil$$

in which $\lceil x \rceil$ means the ceiling of $x$ (the smallest integer number larger than or equal to $x$). This parameter, $n_r$, is also the effective averaging length within the same tolerance $\varepsilon_r$. In other words, if it is desired to have at least $\varepsilon_r$ fraction of a given frame to influence the output of the $n_r$ frames in the future, then the value for $\alpha$ should be:

$$\alpha = e^{\ln(\varepsilon_r)/n_r}$$

In order to analyze the performance of the first-order filter on the input noise, some statistics of the noise process should be assumed. For example, if the input to the filter is a zero mean stationary noise sequence with known auto-correlation function $R_n(\ell)$, defined by:

$$R_n(\ell) = \mathrm{E}\{x(k)x(k-\ell)\}$$

The output would also be a zero mean stationary sequence with its auto-correlation function given by

$$R_y(\ell) = \left(\frac{1-\alpha}{1+\alpha}\right)\alpha^{|\ell|} * R_n(\ell)$$

in which $*$ stands for discrete linear convolution. When the input noise is white with the variance of $\sigma_n^2$, the output auto-correlation function becomes

$$R_y(\ell) = \sigma_n^2 \left(\frac{1-\alpha}{1+\alpha}\right)\alpha^{|\ell|}$$

which results in

$$\sigma_y^2 = R_y(0) = \sigma_n^2 \left(\frac{1-\alpha}{1+\alpha}\right)$$

In this case the ratio of the output power over the input power is simply given by:

$$\frac{\sigma_y^2}{\sigma_n^2} = \left(\frac{1-\alpha}{1+\alpha}\right)$$

Apparently as $\alpha \to 1$ the filtering of the noise approaches perfection. However, as $\alpha \to 0$ the system approaches zero noise filtering.

Proper selection of $\alpha$, for a non-adaptive stationary case with no motion detection, should be based on a given objective. The effective number of frame averaging, $n_r$, as suggested above is a good objective. For

Table 1: First-Order Low-Pass Noise Filtering Characteristics (note that $\left(\sigma_y^2/\sigma_n^2\right)$ are specified in dB).

| | $\varepsilon_r = 0.1$ | | $\varepsilon_r = 0.05$ | | $\varepsilon_r = 0.01$ | |
|---|---|---|---|---|---|---|
| $n_r$ | $\alpha$ | $\left(\sigma_y^2/\sigma_n^2\right)$ | $\alpha$ | $\left(\sigma_y^2/\sigma_n^2\right)$ | $\alpha$ | $\left(\sigma_y^2/\sigma_n^2\right)$ |
| 0 | 0.0 | 0.00 | 0.0 | 0.00 | 0.0 | 0.00 |
| 1 | .10 | -0.87 | .05 | -0.43 | .01 | -0.09 |
| 2 | .32 | -2.84 | .22 | -1.98 | .10 | -0.87 |
| 3 | .46 | -4.37 | .37 | -3.36 | .22 | -1.90 |
| 4 | .56 | -5.53 | .47 | -4.46 | .32 | -2.84 |
| 5 | .63 | -6.45 | .55 | -5.36 | .40 | -3.66 |
| 6 | .68 | -7.22 | .61 | -6.12 | .46 | -4.37 |
| 7 | .72 | -7.88 | .65 | -6.76 | .52 | -4.98 |
| 8 | .75 | -8.45 | .69 | -7.33 | .56 | -5.53 |
| 9 | .77 | -8.95 | .72 | -7.83 | .60 | -6.01 |
| 10 | .79 | -9.41 | .74 | -8.28 | .63 | -6.45 |
| 11 | .81 | -9.82 | .76 | -8.69 | .66 | -6.85 |
| 12 | .83 | -10.2 | .78 | -9.06 | .68 | -7.22 |
| 13 | .84 | -10.5 | .79 | -9.40 | .70 | -7.56 |
| 14 | .85 | -10.9 | .81 | -9.72 | .72 | -7.88 |
| 15 | .86 | -11.2 | .82 | -10.0 | .74 | -8.17 |
| 16 | .87 | -11.4 | .83 | -10.3 | .75 | -8.45 |

different values of this parameter, and $\alpha$, the amount of noise filtering is specified in Table 1.

Similar discussion can be carried out for the second order filter given in (6). In this case, the filter's transfer function is

$$H(z) = \frac{(1-\alpha)^2}{(1-\alpha z^{-1})^2}$$

and its impulse-response is given by

$$h(k) = (1-\alpha)^2(k+1)\alpha^k u(k)$$

The step-response of this filter takes longer time (more number of frames) to settle; i.e., has longer rise-time. In this formulation, the relation between $\varepsilon_r$, $n_r$, and $\alpha$ is implicit and is given by:

$$(n_r + 1 - \alpha n_r)\alpha^{n_r} = \varepsilon_r$$

By using numerical methods, characteristic of this filter under different conditions are summarized in Table 2.

Auto-correlation function of the output can be related to that of the input by:

$$R_y(\ell) = \left(\frac{1-\alpha}{1+\alpha}\right)^2 \alpha^{|\ell|} * \alpha^{|\ell|} * R_n(\ell)$$

When the input is a zero mean stationary white noise with variance of $\sigma_n^2$, the output power (variance) is

Table 2: Second-Order Low-Pass Noise Filtering Characteristics (note that $\left(\sigma_y^2/\sigma_n^2\right)$ are specified in dB).

| | $\varepsilon_r = 0.1$ | | $\varepsilon_r = 0.05$ | | $\varepsilon_r = 0.01$ | |
|---|---|---|---|---|---|---|
| $n_r$ | $\alpha$ | $\left(\sigma_y^2/\sigma_n^2\right)$ | $\alpha$ | $\left(\sigma_y^2/\sigma_n^2\right)$ | $\alpha$ | $\left(\sigma_y^2/\sigma_n^2\right)$ |
| 0 | 0.0 | 0.00 | 0.0 | 0.00 | 0.0 | 0.00 |
| 1 | .05 | -0.87 | .03 | -0.43 | .01 | -0.09 |
| 2 | .20 | -3.11 | .14 | -2.21 | .06 | -0.99 |
| 3 | .32 | -4.88 | .25 | -3.87 | .14 | -2.29 |
| 4 | .42 | -6.18 | .34 | -5.18 | .22 | -3.49 |
| 5 | .49 | -7.18 | .42 | -6.20 | .29 | -4.51 |
| 6 | .55 | -7.99 | .48 | -7.04 | .36 | -5.37 |
| 7 | .59 | -8.67 | .53 | -7.73 | .41 | -6.09 |
| 8 | .63 | -9.26 | .57 | -8.33 | .46 | -6.72 |
| 9 | .66 | -9.77 | .61 | -8.86 | .50 | -7.26 |
| 10 | .69 | -10.23 | .64 | -9.32 | .53 | -7.75 |
| 11 | .71 | -10.64 | .66 | -9.74 | .56 | -8.18 |
| 12 | .73 | -11.0 | .68 | -10.1 | .59 | -8.58 |
| 13 | .75 | -11.4 | .70 | -10.5 | .61 | -8.93 |
| 14 | .76 | -11.7 | .72 | -10.8 | .63 | -9.27 |
| 15 | .78 | -12.0 | .74 | -11.1 | .65 | -9.57 |
| 16 | .79 | -12.3 | .75 | -11.4 | .67 | -9.86 |

given by:

$$\begin{aligned}\sigma_y^2 &= R_y(0) = \sigma_n^2 \left(\frac{1-\alpha}{1+\alpha}\right)^2 \left(\frac{1+\alpha^2}{1-\alpha^2}\right) \\ &= \sigma_n^2 \cdot \frac{1-\alpha+\alpha^2-\alpha^3}{1+3\alpha+3\alpha^2+\alpha^3}\end{aligned}$$

In this case, the ratio of the output power over the input power is simply given by:

$$\frac{\sigma_y^2}{\sigma_n^2} = \frac{1-\alpha+\alpha^2-\alpha^3}{1+3\alpha+3\alpha^2+\alpha^3}$$

This relation has similar properties as before, namely; as $\alpha \to 1$ the filtering of the noise approaches perfection and as $\alpha \to 0$ the system approaches zero noise filtering.

Proper selection of $\alpha$ for non-adaptive cases can be based on a desired value of $n_r$. For different values of this parameter, and $\alpha$ the amount of noise filtering is specified in Table 2. By careful study of Tables 1 and 2, we can see how parameter $\alpha$ affects performance of the two filters and how the two different filters are compared against each other for various $\varepsilon_r$'s.

For example, for the first-order filter, if it is desired to have the maximum effective averaging to be about 16 frames, in the case of 30 frames per second, $\alpha$ should be bounded to the maximum of $0.75$ in order to reduce the effect of the frames that are farther than

16 away from the present one to less than 1%. For the second-order filter this upper bound should be set at about 0.67. In the first-order filter case; (5), the noise amplitude will be reduced by about 62% (about 8.45 dB) when $\alpha = 0.75$ and in the case of the second-order filter; (6), by about 68% (about 9.86 dB) for $\alpha = 0.67$.

When there is a motion, we would like to reduce the effective averaging length in time so that the motions appear with minimal lagging. In this situation, ideally, depending on the statistics of the motion, its speed of change, the temporal averaging should be adjusted. Motion characteristic changes from one pixel to another and any adjustment of the effective averaging length should be conducted adaptively. This issue is discussed in the next section. Adaptive adjustment of this parameter, for optimum noise filtering, is desirable under all conditions. In the next section, the general adaptation process and the corresponding issues are discussed and the problem of automatically adjusting $\alpha$ is addressed.

### 3.4 Adaptive Filtering

In general most adaptive filters are based on adaptively adjusting the parameters of the supposedly optimum filter based on estimation of the unknown parameters. For example, adaptive Wiener or Kalman filter can be based on an on-line estimation of the signal and noise statistics from available data. These algorithms, in general, are computationally intense and expensive. With some minor assumptions, the adaptation can be made more affordable. For example, if it is assumed that the noise is an independent Gaussian random process with zero mean but unknown variance, or slowly varying variance (both in time and space), then the unknown variance can be estimated by using the residual of the filter over the last N samples. Or when the signal is not stationary, it can be assumed that it is locally stationary and the local data can be used to estimate the signal statistics; i.e., autocorrelation function.

The general problem of adaptive filters is discussed in the work by [7]. In particular, some relevant algorithms for adaptive spatial-temporal noise filtering for video images are reviewed by [2]. In this case, the adaptation is done in terms of first estimating what pixels, both in space and time, belong to an object and then only use them for averaging. The problem of estimation of pixels of a spatial-temporal object is computationally very expensive and does not lend itself to the real-time processing. However, if the problem is divided into two independent problems of adaptation in the spatial domain and adaptation in the temporal domain, then the computational expense is more manageable.

Adaptation in the spatial domain may require edge detection and segmentation of the local pixels. In this case, for any given pixel and its local surroundings, a decision has to be made as to what region it belongs to and only use pixels in that region to estimate the unknown parameters (or conduct the averaging only over those pixels). Another, simpler approach in this case, is to adaptively estimate the local statistics, with no regards to the edges and segments. In this case, with reference to (4), $h_{oh}$ and $h_{ov}$ can be calculated for each pixel by using the local neighboring pixels to estimate $R_{nh}$, $R_{nv}$, $r_{dh}$, and $r_{dv}$. When there is an edge in the local pixels, the variance estimate increases and as a result strength of the filtering will be reduced.

Although, there are several computationally efficient algorithms for adaptive calculation of $h_{oh}$ and $h_{ov}$, but due to the real-time requirement and inverse matrix calculation at each pixel, these adaptive algorithms are not employed for real-time processing. One common approach in spatial filtering of images is to employ nonlinear filters like order-statistic and cluster filtering which are robust with respect to edges. Although these filters are computationally more expensive than linear filtering, however, they are more affordable than previously mentioned adaptive spatial filters.

Adaptation in the temporal direction can be done in two ways. One approach is based on the estimation of motion and the other one is based on the estimation of the short-term temporal variance. In each case, when the motion is present or when the temporal variance increases, the filtering is reduced to prevent the lagging effect in the motion. Nonlinear temporal filters are more robust to the motion and lagging effect. In the remaining part of this section only temporal adaptation, in terms of variance estimation and motion estimation, is discussed.

In our discussion of adaptive temporal filter the following model is used for any pixel in a discrete time. Let a given pixel be represented by $x(k)$, where $k$ represents the discrete time index or frame index. The ideal unknown noise-free pixel value is assumed to be $s(k)$, with unknown second-order statistics. It is assumed that the signal is a first-order autoregressive (AR) model. Other orders or models can be assumed, but the first-order AR model is the more realistic and simple model to work with. Under this assumption, the process and noise model equations are:

1. $s(k+1) = as(k) + w(k)$, where $a$ is a constant that depends on the signal statistics and $w(k)$, is the process noise, assumed to be a zero-mean white Gaussian random process with variance of

$\sigma_w^2$.

2. $x(k) = s(k) + v(k)$ is the noise measurement of the signal and $v(k)$ is the independent additive zero mean Gaussian white noise with variance of $\sigma_v^2$.

Here it is implicitly assumed that the noise and signal are stationary random processes that are fully determined by their second-order statistics. Although this assumption is not completely true, but we can use it in practice for local statistics. With above model and assumptions, our approach for filtering noise would be the recursive discrete Kalman filter. In this approach, the recursive algorithm is conducted based on the following definitions:

1. $y(k)$, is the output of the filter, estimate of the signal, at time $k$.

2. $\sigma^2(k) = \mathrm{E}\{[y(k) - s(k)]^2\}$, variance of the estimation error, initially unknown. It should be pointed out that this variance is also representing the level of the noise after filtering. Optimum value of this parameter is achieved only in the steady-state condition and for stationary signal.

3. $K(k)$, is used to represent the Kalman filter gain.

The Kalman filter algorithm is then given as follows:

Let $y(-1) = 0,$ and $\sigma^2(-1) = \sigma_v^2,$

Start by setting $k = 0$

**LOOP:** for $k$ do the following:

$$K(k) = \frac{a^2\sigma^2(k-1) + \sigma_w^2}{a^2\sigma^2(k-1) + \sigma_w^2 + \sigma_v^2}$$

$$y(k) = K(k)x(k) + a[1 - K(k)]y(k-1)$$

$$\sigma^2(k) = a^2[1 - K(k)]\sigma^2(k-1) + \sigma_w^2$$

If finished go to **END** otherwise:

Increment $k$, $k \leftarrow k + 1$, and go to **LOOP**

**END**

In this algorithm, there are couple of parameters that are assumed to be known but in practice are unknown. These parameters are, $\sigma_v^2$, $\sigma_w^2$, and $a$. The algorithm can be made adaptive by properly estimating these parameters, using local available data. Estimation of these parameters can be based on optimization of a criterion function like minimum mean-square error (MMSE). No matter which estimation technique is used, the quality or reliability of the estimates, will always depend on the length of data used to estimate them. Based on the aforementioned assumptions, the following instantaneous estimates, in each step of the recursive Kalman filter, can be used to achieve a fast and simple implementation:

$$a = \frac{\mathrm{E}\{x(k)x(k-1)\}}{\mathrm{E}\{x^2(k)\}} \cong \frac{x(k)x(k-1)}{\frac{1}{2}[x^2(k) + x^2(k-1)]}$$

It is suggested to use some *a priori* information about the signal and keep this parameter constant for stability issues.

$\sigma_v^2 = \mathrm{E}\{[x(k) - ay(k-1)]^2\} \cong [x(k) - ay(k-1)]^2$

$\sigma_w^2 = \mathrm{E}\{[x(k) - \hat{a}y(k-1)]^2\} \cong [x(k) - \hat{a}y(k-1)]^2$

These simple procedures are very noise sensitive and should be tested under practical circumstances for their usability. The advantage of an adaptive over a non-adaptive method is that, in the adaptive case, the Kalman gain starts with relatively large values, and gradually decreases when the temporal signal is stationary. This is a desired property which produces more noise filtering as time goes by. However, when there is a motion, the estimate of the noise variance, $\hat{\sigma}_v^2$, and process noise variance $\hat{\sigma}_w^2$, increase and cause the Kalman gain to increase and consequently, the filter output dependency be weighted more on the present signal than previous output. This will reduce the noise filtering so that it can better follow the motion with a minimal lagging effect.

In the above adaptation, only two sample points (from two consecutive frames) are used to estimate variance and correlation. As a result, the filter output is relatively noisy and the overall filtering is not in the desired optimum level. An example of variation of a single pixel value in discrete time (from one frame to another) is simulated and shown in Figure (3). In this example, there is a sudden step change in the signal, representing motion, with the signal-to-noise ratio ($SNR$) of 20 dB. The smooth solid line in this example is the result of the non-adaptive Kalman filter using true parameter values. As it is evident, the non-adaptive filter completely goes out of its desired behavior when there is a sudden change in the signal. The adaptive filter is robust with respect to the sudden change but is not optimum in terms of the noise filtering.

If storage of more frames can be afforded, then a better adaptation can be employed by using more data points for estimation of the unknown parameters. Better estimates of the unknown parameters may provide a better filtering result but generates more lagging effect in the order of the number of frames used for parameter estimation. In Figure (4), an example is given in which eight frames are used for parameter estimation. From this example, it is evident that adaptation is not improved significantly while an eighth-order lagging is introduced.

The above results show the typical behavior of an adaptive Kalman filter. A better approach, in this case,

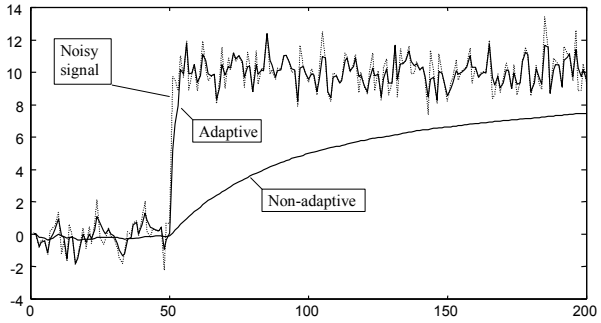


Figure 3: Filtering a noisy step-like signal, representing motion caused change, using optimum non-adaptive and adaptive Kalman filter (in which $a$ is kept constant for stability).

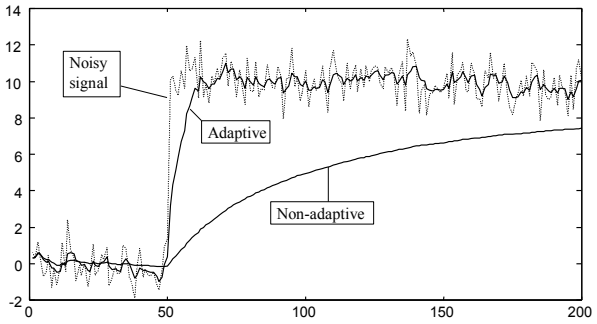


Figure 4: Filtering a noisy step-like signal using optimum non-adaptive and adaptive Kalman filter with eight sample storage (in which $a$ is kept constant for stability).

would be to assume that the exact values of these parameters are known and only try to estimate the location of the sudden changes, associated to a motion in the image sequence. This approach also requires more frame storage for motion estimation and generates a lagging effect proportional to the number of frames used to estimate the motion (assuming that the motion is correctly estimated).

One example for motion estimation is to compare two consecutive temporal samples and use their magnitude difference to infer existence or nonexistence of motion. This estimation can be conducted by using a statistical test based on the assumption of white Gaussian noise; [10]. For example, when there is a sudden change in the signal, the difference between $ay(k-1)$ and $x(k)$ goes beyond its statistical variation, with some level of statistical confidence. In this case, a simple test can be employed to estimate sudden changes in the signal (motion in the image sequence). Assuming that $\sigma_v^2$ represents the variance of the aforementioned differences, then based on Gaussian noise distribution, it can be said that a motion is present if

$$\gamma = \frac{|x(k) - ay(k-1)|}{\sigma_v} \geq \Gamma$$

If the test is positive, then the gain calculation in the Kalman filter algorithm can be re-initiated by assuming $\sigma^2(k) = \sigma_v^2$. This new gain value significantly reduces the lagging effect while improves the noise filtering. There are other modifications that can be used. For example, the Kalman gain can directly be set equal to 1 and the filtering be re-initiated right after the motion. Or both $\sigma^2(k)$ and $\sigma_w^2$ can be set equal to $\sigma_v^2$, which results the Kalman gain to be re-initiated to

$$K(k) = \frac{a^2+1}{a^2+2}$$

which would be equal to a relatively large value of $2/3$ for $a = 1$.

Table 3: Threshold values for Motion detection

| Confidence Level | $\Gamma^2$ | $\Gamma$ |
|---|---|---|
| 99.90% | 10.827 | 3.29 |
| 99% | 6.635 | 2.576 |
| 98% | 5.412 | 2.326 |
| 95% | 3.841 | 1.96 |
| 90% | 2.706 | 1.645 |

Selection of the threshold $\Gamma$, is very important. In this case, it can be shown that $\gamma^2$ has $\chi^2$ distribution with one degree of freedom. For the purpose of statistical test of hypothesis, different values of $\Gamma$, for different confidence level, are tabulated in Table 3.

Proper use of these values will result in the same confidence level as stated in Table 3. In other words, if the noise is white and independent from signal, using 95% confidence will produce only 5% error, in average, when it is used for motion detection. In Figure (5), the Kalman filter with motion detection is compared with the standard non-adaptive optimum Kalman filter when the $SNR = 20$ dB. In this simulation, the confidence level is kept at the highest level given in Table 3; namely 99.9%, which corresponds to $\Gamma = 3.29$.

Similar simulation is conducted on a signal with $SNR = 10$ dB. In this case, we have used two different confidence level; one at the highest level in Table 3 and the other one at the level of 95%. Figure (6) depicts the result for 99.9% confidence level and Figure (7) depicts the result for 95% confidence level

In these examples, it is assumed that $\sigma_v^2$ is known and the selected values for thresholds are simply based

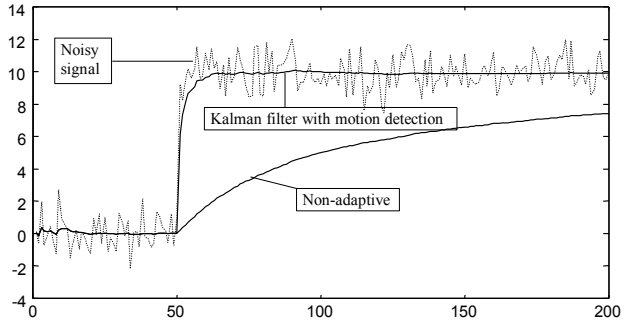


Figure 5: The result of Kalman filtering, using motion detection, in comparison with the non-adaptive filtering when $SNR = 20$ dB and $\Gamma = 3.29$.

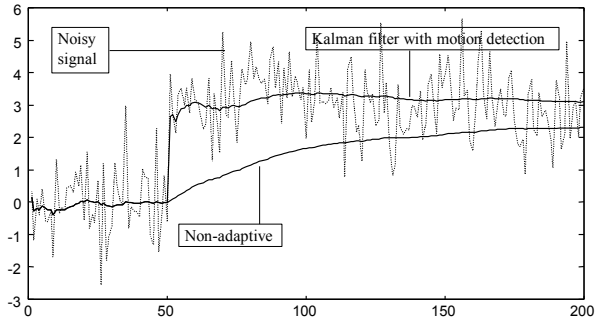


Figure 6: The result of Kalman filtering, using motion detection, in comparison with the non-adaptive filtering when $SNR = 10$ dB and $\Gamma = 3.29$.

on a desired confidence level. In any given application, the user should provide a reasonable value for $\sigma_v^2$ based on his or her *a priori* information. As it is evident, the filter is performing very good in the case of high confidence level and not very good for low $SNR$ and lower confidence level. It should be emphasized that this approach is sensitive to impulsive noise and assigns an impulsive noise to a motion. Therefore, not only does not filter the impulsive noise but also restarts the filter and deviates from its optimum operation. In the case of impulsive noise, an example with $SNR = 20$ dB and $\Gamma = 3.29$, is simulated and the results are shown in Figure (8). This particular problem of impulsive noise can be minimized by using some form of local spatial filtering before temporal motion detection. In other words, instead of using original frames, we can use spatially filtered version of that frame sequence just for the purpose of motion detection.

In all cases, when using Kalman filter, the level of noise after filtering at each time can be monitored by reading out the values for $\sigma^2(k)$. In the stationary case, where there is no motion (sudden changes in

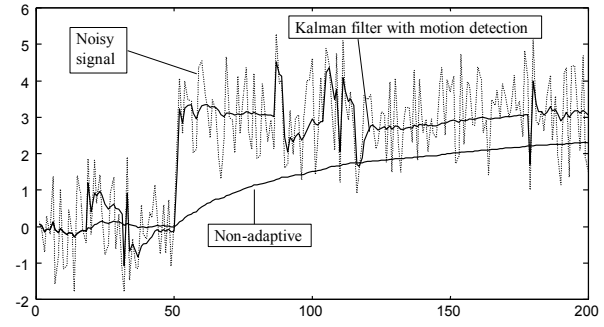


Figure 7: The result of Kalman filtering, using motion detection, in comparison with the non-adaptive filtering when $SNR = 10$ dB and $\Gamma = 1.96$.

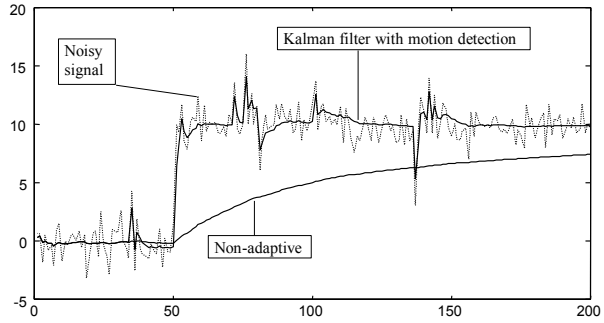


Figure 8: The result of adaptive Kalman filtering, using motion estimation, in comparison with the non-adaptive filtering when $SNR = 20$ dB and $\Gamma = 3.29$. In this case there are 10% impulsive noise present.

the signal), this value approaches its optimum value. For the adaptive case, or when motion detection is employed, as long as the signal remains stationary, $\sigma^2(k)$ approaches its optimum value, but right after detection of a motion or changes in the adaptive parameters, the noise filtering is reduced and $\sigma^2(k)$ will increase to incorporate for that change. This is one of the interesting and important feature of the Kalman filter.

In cases where spatially smoothed version of each frame is available, the motion estimator can become less sensitive to noise by using the smoothed frame instead of the original frame in the motion estimation process. This approach will improve the performance of the adaptive filter for lower $SNR$. The problem of impulsive noise can also be resolved by using some form of nonlinear filtering approach, some of which are discussed in the following sub-section.

### 3.5 Nonlinear Filtering

The class of nonlinear filters that are of interest in image processing is the group of filters that are robust

with respect to sudden changes (edges in spatial domain and motions in temporal domain) in the signal. The first proposed filter in this class, and the simplest one, is the median filter. Generalized version of this filter is the order-statistics filter and stack filters; [14]. Different modifications and adjustments are proposed which address some of the issues concerning the computational complexities and performance of these filters; [1].

Another class of nonlinear filters is the one based on cluster analysis and fuzzy set theory; [5]. These filters, generally, require extensive computations and are employed only on still images. The simplest nonlinear filter, namely median filter, can be used for real time processing by designing special hardware, VLSI, that is based on the idea of stack filters to calculate the median of a set of integer numbers. In this subsection, due to the interest for image sequence application, only spatial median filter and temporal median filter are discussed.

In spatial median filter, a 2-D moving window scans each frame, one pixel at a time. At each position, all the pixels in the window are used to find the median as the filter output. This approach, as will be shown for the 1-D temporal filter, is robust to impulsive noise and preserves sudden changes (edges) in the image. This filter (and many of its variations) is readily used on images to maximally remove impulsive noise. Independent Gaussian noise will also be removed, but not as much as it is possible with the optimum filter. The 2-D filter size is selected based on the area of the largest undesirable impulsive noise or based on the smallest desirable detail in the image. If the area of the largest undesirable impulsive noise is $n_i$ pixels, then the window size should be at least $|\sqrt{2n_i+1}| \times |\sqrt{2n_i+1}|$, or if the smallest detail of interest in the image is of $n_d$ pixels, then the window size should be at most equal to $|\sqrt{2n_d+1}| \times |\sqrt{2n_d+1}|$. In other words, if the 2-D window, for median filter is $n_w \times n_w$, then $n_w$ should satisfy

$$|\sqrt{2n_d+1}| \geq n_w \geq |\sqrt{2n_i+1}|$$

It should also be pointed out that the larger the window size the better the noise filtering would be. Therefore, the upper limit of this inequality would be a better choice for median filtering of the image in the spatial domain.

In the temporal domain, median filter can be used to preserve motions while filtering noise. Although noise filtering is not optimal, but most of impulsive noises would be removed without any lagging effect in the image sequence. The size of the temporal median filter is selected similar to the 2D spatial median filter as follows. If the temporal length of an impulsive noise is $n_i$ frames (usually we can assume that $n_i$ is equal to 1) and the maximum effective averaging in time is $n_e$ frames, then the length of the temporal median filter is obtained from

$$n_e \geq n_w \geq 2n_i + 1$$

In general, the maximum possible size, limited by hardware complexity, is used to improve the noise filtering. The minimum possible size would be 3. In the following figures, i.e., Figures (9), (10), and (11), several examples are presented for demonstration.

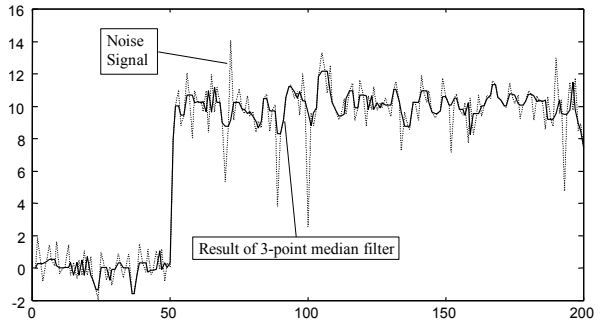


Figure 9: Result of 3-point median filter on a signal with $SNR = 20$ dB and 10% impulsive noise.

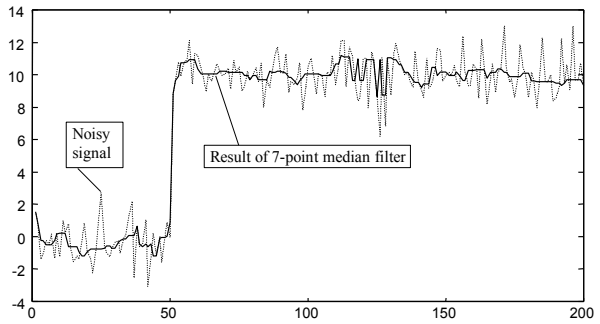


Figure 10: Result of 7-point median filter on a signal with $SNR = 20$ dB and 10% impulsive noise.

As it can be seen from these simulated results, the median filter is robust with respect to impulsive noise and somewhat filters the Gaussian noise as well. As the window size increases, the filtering of Gaussian noise becomes more significant. The main characteristic of the median filter, as it is evident from these examples, is the preservation of steps, which relates to motion in an image sequences. From implementation point of view, this approach requires $n_w$ storage frames and VLSI implementation of median filters. In the next section we address issues related to the implementation of the algorithms discussed in this work.

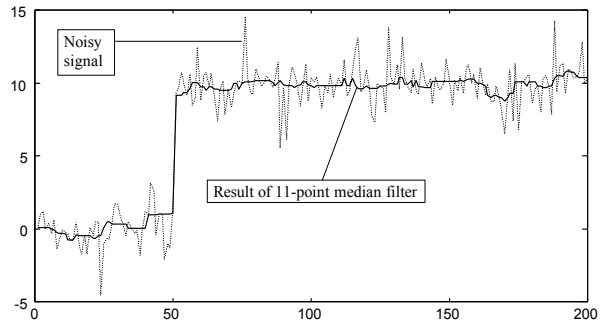


Figure 11: Result of 11-point median filter on a signal with $SNR = 20$ dB and 10% impulsive noise.

# 4 Real-Time Computation and Implementation Issues

One of the important issues in the real-time image sequence processing is the hardware implementation of the algorithms. The linear 3D filtering presented in (2) requires storage for enough number of frames for proper spatial-temporal filtering. If the 3D filter is L-th order in the temporal direction then there is a storage need for at least L frames. In terms of the number of operations, assuming that each frame is $N \times N$ and there are 30 frames per second, the order of operations (multiplications/additions) is about $30N^2M^2L$ operations per second, where $M \times M$ is the size of the spatial FIR filter and $L$ is the order of the temporal filter. For example, for $N = 1024$, $M = 11$,and $L = 5$, there will be a need for $20 \times 10^9$ multiplications/additions per second. When $N = 512$, $M = 5$, and $L = 3$, the order of operations reduces to about $6^8$ operations per second, which is still at a very high formidable rate for real-time processing. Even with todays technology, this kind of processing can only be carried out using dedicated VLSI/FPGA hardware design.

When we only use spatial filters for noise removal; i.e., (4), our storage need is at least one frame and the number of operations per second will be about $30N^2M^2$. For example, for $N = 1024$ and $M = 11$, there will be a need for $4 \times 10^9$ multiplications/additions per second. When $N = 512$ and $M = 5$, the order of operations reduces to about $2^8$ operations per second, which is still highly formidable.

For the temporal-only fixed non-adaptive filtering proposed in (5) and (6), our storage need is between two to five frames and the number of operations is about $60N^2$ per second for the first-order system and $120N^2$ for the second-order filter. When $N = 1024$ the order of operation is about $64 \times 10^6$ operations per second for the first-order filter and $128 \times 10^6$ operations per second for the second-order filter, which is completely manageable with todays high end processors. We can still be more efficient by using VLSI design or hardware specific design. These rates will be reduced by a factor of 4 when the image size is $N = 512$.

Implementation of the Kalman temporal filter requires at least five storage frames and about 10 operations (multiplications/additions) per pixel, which translates to $300N^2$ operations per second. For $N = 1024$ we get the order of operations to be about $320 \times 10^6$ operations per second. When $N = 512$ this rate is decreased by the factor of 4 to $80 \times 10^6$ operations per second. This rate is manageable with todays high-end processors and particularly with dedicated VLSI/FPGA design.

Implementation of the nonlinear, median, temporal filter requires about 3 to 5 frames of storage respectively for 3-point and 5-point median filtering. Number of comparison for sorting and finding the median is at most 3 for 3-point median and at most 4 for 5-point median operation. Total number of comparisons per second for $N = 1024$ is about $160 \times 10^6$ per second for 5-point median and $95 \times 10^6$ for 3-point median. This rate is within the acceptable range of operations and can be handled with todays technology.

Based on the examples and discussions presented thus far, several recommendations are made in the following conclusion section.

# 5 Conclusion

In this article, the problem of noise filtering in image sequences is discussed and for some limited cases, several simulation results are presented. Based on presented examples, we recommended using the presented Kalman filter along with the motion detection algorithm. Although it is possible to use a second-order recursive filter for better peformance, but as it is shown here, in Tables 1 and 2, the difference between the two, for a given effective averaging length, is at most in the order of 2 dB. Adaptation and motion detection for the second order recursive filter, however, would be more complex.

The first-order Kalman filter with motion detection algorithm, in a simple form, would be as given in the following frame. If there exists an impulsive noise in the image sequence, instead of original frames, use spatially filtered frames. The only parameters that should be supplied by the user are $\sigma_v^2$, variance of the white noise, and $\Gamma$; the desired confidence level for the motion detection. If the noise is signal dependent, then for motion detection there is a need to properly normalize the $D$ parameter in the algorithm. The only

issue that should be resolved is how to use the two new parameters, $\sigma^2$ and $\sigma_w^2$, as explained in the algorithm. For further simplification, with some approximation, $\sigma_w^2$ can be eliminated from the algorithm with a minor adjustment in the update equation for $\sigma^2$.

Let $y(-1) = 0, \sigma_w^2 = \sigma_v^2,$ and $\sigma^2 = \sigma_v^2,$

Start by setting $k = 0$

**LOOP:** for $k$ do the following:

$$K = \frac{\sigma^2 + \sigma_w^2}{\sigma^2 + \sigma_w^2 + \sigma_v^2}$$

$$y(k) = Kx(k) + (1 - K)y(k - 1)$$

if $D = \dfrac{|x(k) - y(k-1)|}{\sigma_v} \geq \Gamma$

$\sigma_w^2 = \sigma_v^2$

$\sigma^2 = \sigma_v^2$

else

$\sigma_w^2 = K^2\sigma_v^2$

$\sigma^2 \leftarrow (1 - K)\sigma^2 + \sigma_w^2$

end-if

If finished go to **END** otherwise:

Increment $k$, $k \leftarrow k + 1$, and go to **LOOP**

**END**

For future development, new approaches should be considered, both for filtering and motion estimation. These approaches should include median filtering, cluster filtering, and wavelet decomposition of images. The next step, in improving the temporal filter, would be the use of temporal median filter, in real time, not for the filtering purposes, but for the motion detection. This of course will increase the hardware requirement for the temporal filtering.

## DISCLAIMER AND NOTE

The views expressed herein are those of the author and are not to be construed as official or reflecting the views of the Commandant, the U.S. Coast Guard, the Department of Homeland Security, or any agency of the U.S. Government.

*References:*


[1] Astola, J., Haavisto, J., and Neuvo, Y., "Vector median filters." Proceedings of The IEEE, vol. 78, no. 4, April 1990.

[2] Braileam, J.C., Kleihorst, R. P., Efstratiadis, S., Katsaggelos, A. K.,and Lagendijk, R. L., "Noise reduction filters for dynamic image sequences: A review." Proceedings of The IEEE, vol. 83, no. 9, September 1995.

[3] Brown R. G. and Hwang, P. Y. C., Introduction to Random Signals and Applied Kalman Filtering, 3rd Ed., John Wiley and Sons, New York, 1997.

[4] Chin, R. T. and Yeh, C. L., "Quantitative evaluation of some edge-preserving noise-smoothing techniques." Computer Vision, Graphics, and Image Processing, vol. 23, pp. 67-91, 1983.

[5] Doroodchi, M. and Reza, A. M., "Fuzzy cluster filter." Proceedings of The 3rd International Conference on Image Processing in ICIP96, September 1996, Lausanne, Switzerland.

[6] Doroodchi, M. and Reza, A. M., "Implementation of fuzzy cluster filter for nonlinear signal and image processing." Proceedings of The 5th IEEE International Conference on Fuzzy Systems, FUZZ-IEEE, September 1996, New Orleans, LA.

[7] Haykin, S., Adaptive Filter Theory, 3rd Ed., Prentice Hall, New Jersey, 1996.

[8] Hou, Sheng-Yun, Hung, Hsien-Sen, Chang, Yuan-Chang, and Change, Shun-Hsyung, "Multitarget Tracking Algorithms Using Angle Innovations and Extended Kalman Filter," WSEAS Transactions on Systems, vol. 8, no. 3, March 2009.

[9] Kassam, S. A. and Poor, H. V., "Robust techniques for signal processing: A survey." Proceedings of The IEEE, vol. 73, no. 3, March 1985.

[10] Kirk, R.E., Statistics, An Introduction, 3rd Ed., Holt, Rinehart and Winston, Inc., Texas, 1990.

[11] Reza, A. M. and Doroodchi, M., "Cramer-Rao Lower Bound on Locations of Sudden Changes in a Step-Like Signal," IEEE Transactions on Signal Processing, vol. 44, no. 10, October 1996.

[12] Tozan, Hakan and Yagimli, Mustafa, " Fuzzy Prediction Based Trajectory Estimation," WSEAS Transactions on Systems, vol. 9, no. 8, February 2010.

[13] Volosencu, Constantin, "Properties of Fuzzy Systems," WSEAS Transactions on Systems, vol. 8, no. 2, February 2009.

[14] Wendt, P. D., Coyle, E. J., and Gallagher, N. C. Jr., "Stack filters." IEEE Transactions on Acoustics, Speech, and Signal Processing, vol. ASSP-34, no. 4, August 1986.